AMENDMENTS TO THE SPECIFICATION

At page 9, beginning at line 9 and through line 25, replace the two paragraphs with the following two paragraphs to correct typographical errors.

Turning now to Figs. 12-14, two laptop computers 101 and 102 are illustrated. The computers 101 and 102 are not necessarily identical and indeed, one can be an IBM-clone and the other can be an APPLE computer. The computers 101 and 102 have two screens 105, 115 two keyboards 106, 116 but a single mouse 107. The two machines 101, 102 are interconnected by a means of a single coaxial cable or twisted pair cable 114.

Two simple application programs are downloaded onto each of the machines 101, 102, the programs being modified as they are being loaded as described above. In this embodiment the first application is a simple calculator program and results in the image of a calculator 108 being displayed on the screen 105. The second program is a graphics program which displays four coloured blocks 109 which are of different colours and which move about at random within a rectangular box 110. Again, after loading, the box 110 is displayed on the screen 105. Each application operates independently so that the blocks 109 are in random motion on the screen 105 whilst numerals within the calculator 108 can be selected (with the mouse 107) together with a mathematical operator (such as addition or multiplication) so that the calculator 108 displays the result.

Cancel the Annexure that accompanied the application as filed (and follows the drawings in the electronic record of the application with the USPTO) and insert the description provided therein at page 10, after line 8 and before line 9, as follows:

Example Program Listing

Attention is now directed to example program listings of excerpts in the JAVA language.

1.   This first excerpt is part of the modification code.   It searches through the code array, and when it finds a putstatic instruction (opcode 178), it implements the modifications. DRT is a Distributed Run Time.

```
// START
byte[] code = Code_attribute.code;      // Bytecode of a given method in a given classfile.
int code_length = Code_attribute.code_length;
int DRT = 99;     // Location of the CONSTANT_Methodref_info for the DRT.alert()
method.
for (int i=0; i<code_length; i++){
  if ((code[i] & 0xff) == 179){ // Putstatic instruction.
                System.arraycopy(code, i+3, code, i+6, code_length-(i+3));
                code[i+3] = (byte) 184;        // Invokestatic instruction for the
DRT.alert() method.
           code[i+4] = (byte) ((DRT >>> 8) & 0xff);
                code[i+5] = (byte) (DRT & 0xff);
     }
}
// END
```

2.   This second excerpt is part of the DRT.alert() method. This is the body of the
DRT.alert() method when it is called.


```
// START
public static void alert(){
    synchronized (ALERT_LOCK){
            ALERT_LOCK.notify();      // Alerts a waiting DRT thread in the background.
    }
}
// END
```


3.   This third excerpt is part of the Distributed Run Time (DRT) Sending. This code
fragment shows the DRT in a separate thread, after being notified, sending the value across
the network.


```
// START
MulticastSocket ms = DRT.getMulticastSocket();      // The multicast socket used by the
DRT for communication.
byte nameTag = 33;    // This is the "name tag" on the network for this field.
Field field = modifiedClass.getDeclaredField("myField1");   // Stores the field from the
modified class.
    // In this example, the field is a byte field. while (DRT.isRunning()){
    synchronized (ALERT_LOCK){
            ALERT_LOCK.wait();      // The DRT thread is waiting for the alert method to
be called.
            byte[] b = new byte[]{nameTag, field.getByte(null)};      // Stores the nameTag
and the value of the
        // field from the modified class in a buffer.
            DatagramPacket dp = new DatagramPacket(b, 0, b.length);
```

```
        ms.send(dp);   // Send the buffer out across the network.
    }
}
// END
```

4.   **The fourth excerpt is part of the DRT receiving.   This is a fragment of code to receive a DRT sent alert over the network.**

```
// START
MulticastSocket ms = DRT.getMulticastSocket();    // The multicast socket
used by the DRT for communication.
DatagramPacket dp = new DatagramPacket(new byte[2], 0, 2);
byte nameTag = 33;    // This is the "name tag" on the network for this field.

Field field = modifiedClass.getDeclaredField("myField1");   // Stores the
field from the modified class.
                                                          // In
this example, the field is a byte field. while (DRT.isRunning){
    ms.receive(dp);      // Receive the previously sent buffer from the network.
    byte[] b = dp.getData();
    if (b[0] == nameTag){       // Check the nametags match.
        field.setByte(null, b[1]);    // Write the value from the network packet into the
field location in memory.
    }
}
// END
```